

A Self-synchronized Image Encryption Scheme

Amir Daneshgar¹ and Behrooz Khadem

Sharif University of Technology - Department of Mathematical Sciences

P.O. Box 11155-9415, Tehran, Iran.

`daneshgar@sharif.ir`

Kharazmi University - Faculty of Mathematics and Computer Science

P.O. Box 15719-14911, Tehran, Iran.

`std_khadem@khu.ac.ir`

Abstract

In this paper, a word based chaotic image encryption scheme for gray images is proposed, that can be used in both synchronous and self-synchronous modes. The encryption scheme operates in a finite field where we have also analyzed its performance according to numerical precision used in implementation. We show that the scheme not only passes a variety of security tests, but also it is verified that the proposed scheme operates faster than other existing schemes of the same type even when using lightweight short key sizes.

KeyWords: chaos, image encryption, self-synchronization.

1 Introduction

Several image encryption schemes have been proposed in the literature based on different approaches for design or implementation, while chaos-based encryption schemes have the advantage of presenting a good combination of speed and security.

It seems that Fridrich [8] is among the first contributors who has proposed an image encryption scheme based on chaotic maps, where in [8] certain invertible chaotic 2D maps on a torus or on a square have been used to create new symmetric block encryption schemes. Many other chaotic image encryption schemes have been proposed ever since with different properties and motivations for application (e.g. see [7, 9, 13, 17, 19] and references therein).

Strictly speaking, one may consider the following challenges when one is trying to design an image encryption scheme (see [16, 19, 20] and references therein):

- The scheme must have a relatively high speed of performance since images usually consist of large blocks of data.
- Since the information content of an image is contained in high frequencies the scheme must possess a high mixing performance.
- According to typical applications, the scheme must be relatively lightweight and should be able to operate with relatively small keys with acceptable security guaranties.
- The scheme must guaranty secure, reliable and fast rates of data transfer.

Considering above facts, chaos-based stream ciphers may seem to be a solution while,

- Although, concentrating on chaotic word-based designs operating in a finite field may seem to be a solution for fast and reliable encryption, one must note that discretizing chaotic maps usually deteriorate their chaotic properties that may lead to weak security conditions.

¹Correspondence should be addressed to `daneshgar@sharif.ir`.

- Data transfer reliability can be achieved using self-synchronization, however, security guaranty is much harder in presence of self-synchronization for the feedback structure.

It seems that one of the main problems with chaotic encryption schemes introduced so far is the direct application of the chaotic sequence which is far from being pseudorandom when it is digitized, which will definitely lead to security weaknesses when the scheme is not design properly (e.g. see [14, 21]). Therefore, to solve the above mentioned and seemingly contradicting challenges, we introduce an image encryption scheme in which we have used a chaotic string indirectly to generate a pseudorandom permutation whose pseudorandomness is guaranteed by the results of [1]. On the other hand to compensate the weakness of discrete permutations in uniformly encrypting the high frequency image data (mainly based on correlation along edges) we use a linear feedback to achieve the acceptable uniformization. In other words we,

- Use pseudorandom permutations generated by chaotic maps.
- Use word-based chaos to guaranty fast encryption.
- Compensate discretization phenomenon using a fast linear feedback.
- Make sure that the scheme can perform in both synchronous and self-synchronous modes by setting parameters, to be able to be used in different channel conditions in a reliable way.
- Make sure that the scheme has a fast receiver as an unknown input observer of the transmitter.

PLCIE ² is an extension of PLC scheme introduced in [12] tuned to be used for image encryption. PLCIE is a word-based chaotic encryption scheme having ℓ -word state vectors that can be controlled by users, giving sufficient flexibility for multi-level security. PLCIE consists of a Initializing phase, internal state update, memory update, encryption and decryption that will be described in detail in Section 2. In Section 3, we apply various tests to verify the performance and the security of the proposed scheme.

2 Description of PLCIE

A digital image usually can be interpreted as a function $z = f(x, y)$ of physical horizontal x and vertical y coordinates, that determine illumination or grayscale value of the picture element (or the pixel) at location (x, y) . A pixel is the smallest addressable element in a display device. The level of illumination at each pixel has a value between 0 and 255. Thus, in a digital image, the grayscale of each pixel is presented by one byte and the whole image is presented by a large matrix of bytes. The histogram of a digital image is a discrete function $h(r_k) = n_k$, where r_k is the k -th gray level and n_k is the number of pixels of the image with gray level r_k .

Let q be a prime power, \mathbb{F}_q be the finite field on q elements³ and $f : \mathbb{R} \rightarrow \mathbb{R}$ be a chaotic map (e.g. as in [6]). Consider a family of maps as $\pi : \mathcal{K} \times \mathbb{F}_q \rightarrow \mathbb{F}_q$ such that for any $k \in \mathcal{K}$ the map $\pi(k, \cdot)$ is a discrete chaotic permutation on \mathbb{F}_q as a discrete approximation of f (e.g. as defined in [3]). The two-variable map π gets a value $k \in \mathcal{K}$ as well as a field element $a \in \mathbb{F}_q$,

²PseudoLinear Chaotic Image Encryption (also see [5] for a switching version and its properties).

³The cryptosystem can be defined on any finite field, however, in our real life applications with a lightweight setup we set $\mathbb{F}_q = GF(16)$ or $\mathbb{F}_q = GF(17)$.

Table 1: Functions used in PLCIE

Title	Form
State update	$\varphi_k : (\mathbb{F}_q^\ell)^2 \times \mathcal{M}^4 \rightarrow \mathbb{F}_q^\ell$
Keystream generator	$\gamma_k : (\mathbb{F}_q^\ell)^2 \times \mathcal{M}^2 \rightarrow \mathbb{F}_q^\ell$
Encryption	$\varepsilon_k : (\mathbb{F}_q^\ell)^2 \times \mathcal{M} \rightarrow \mathbb{F}_q^\ell$
Decryption	$\delta_k : (\mathbb{F}_q^\ell)^2 \times \mathcal{M} \rightarrow \mathbb{F}_q^\ell$
Memory update	$\mu : (\mathbb{F}_q^\ell)^2 \rightarrow \mathbb{F}_q^\ell$

and returns $\pi_k(a) \stackrel{\text{def}}{=} \pi(k, a)$. In 2.1 we will describe how one may compute this family of chaotic permutations.

For all $t \geq 1$, consider $p_t, c_t, z_t \in \mathbb{F}_q$, and let $\langle p_t \rangle, \langle c_t \rangle, \langle z_t \rangle$ be the plain, the cipher, and the keystream sequences in time, respectively. Let ℓ be an integer. Also, define column vectors $\mathbf{p}_t, \mathbf{c}_t, \mathbf{z}_t$ in \mathbb{F}_q^ℓ as

$$\mathbf{p}_t \stackrel{\text{def}}{=} [p_t^{(1)}, 0, \dots, 0]^T, \quad \mathbf{c}_t \stackrel{\text{def}}{=} [c_t^{(1)}, c_t^{(2)}, \dots, c_t^{(\ell)}]^T, \quad \mathbf{z}_t \stackrel{\text{def}}{=} [z_t^{(1)}, z_t^{(2)}, \dots, z_t^{(\ell)}]^T.$$

The internal state $\mathbf{s}_t \in \mathbb{F}_q^\ell$ and internal memory $\tilde{\mathbf{c}}_t \in \mathbb{F}_q^\ell$ are also defined as column vectors

$$\mathbf{s}_t \stackrel{\text{def}}{=} [s_t^{(1)}, s_t^{(2)}, \dots, s_t^{(\ell)}]^T, \quad \tilde{\mathbf{c}}_t \stackrel{\text{def}}{=} [\tilde{c}_t^{(\ell)}, \tilde{c}_t^{(\ell-1)}, \dots, \tilde{c}_t^{(1)}]^T.$$

Define the map $\wp_k : \mathbb{F}_q^\ell \rightarrow \mathbb{F}_q^\ell$ as follows

$$\wp_k([a_1, a_2, \dots, a_\ell]^T) \stackrel{\text{def}}{=} [\pi_k(a_1), \pi_k(a_2), \dots, \pi_k(a_\ell)]^T.$$

PLCIE scheme uses a set of functions introduced in Table 1 in which \mathcal{M} stands for the set of all $\ell \times \ell$ matrices on \mathbb{F}_q . Also PLCIE has a initializing phase along with two other main phases called the kernel computation phase, and the encryption/decryption phase that will be described in what follows.

2.1 The initializing phase

In this phase, a chaotic sequence is produced, that gives rise to the pseudorandom permutation π_k . Also, the initial value vector IV is set according to a uniform distribution. The secret key is a binary string consisting of

- encoding of the system precision $prec$ (one bit indicating 16 or 32 bits representation of numbers).
- encodings of the initial values of the chaotic map (r_0, l_0) , chosen uniformly at random, where $r_0 \in_R (0, 1)$ (presented in $prec$ bits floating point format) and $l_0 \in_R \{1, 2, \dots, 2^{prec} - 1\}$ (presented in $prec$ bits integer format).
- encodings of a number $a \in_R \mathbb{F}_q$ and $(i_1, j_1, e_{i_1 j_1}), \dots, (i_n, j_n, e_{i_n j_n})$, in which $n < \frac{\ell^2}{2}$, indicating an encoding of the matrix \mathbf{E} (to be used later) such that the entries not mentioned in the coding is set to the default value a .

Let ι be a constant integer of order $O(\ell)$ (e.g. for $\ell = 8$ this parameter can be chosen as $\iota = 32$). Then, IV is a 2ℓ word vector which is used to preset $\mathbf{s}_{-\iota}$ and $\tilde{\mathbf{c}}_{-\iota}$. Note that here one may use a random string of length ι as a prefix of plaintext for whitening.

For the chaotic map we have chosen a particular version of the Rényi map [1] with parameter $\beta = 3$ which is defined as follows,

$$\psi(x) = 3x - \lfloor 3x \rfloor, \quad x \in (0, 1). \quad (2.1.1)$$

In [1] the discrete version of this map (called $\psi_d(x)$) is defined as,

$$g(x) = \frac{\lfloor 2^{prec} x \rfloor}{2^{prec}}, \quad \psi_d(x) = 3g(x) - \lfloor 3g(x) \rfloor, \quad x \in (0, 1). \quad (2.1.2)$$

Also it is shown in the same reference that the map has a positive Lyapunov exponent and any of its' successive iterations has acceptable statistical properties. However, we will see that this map by itself is not good enough to be solely used in an image encryption scheme as a pseudorandom source for permutations (see Section 2.1).

After producing the iterated sequence $\{\psi_d^t(r_0)\}_{t=1}^{l_0+q}$, and eliminating the first l_0 transient elements, the sequence is used to generate a pseudorandom permutation.

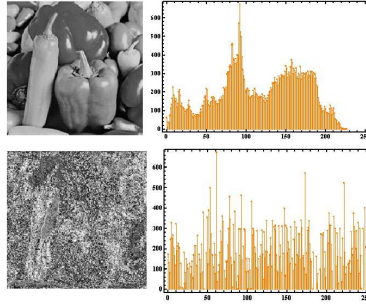


Figure 1: Peper original versus value-permuted image.

The initializing algorithm gets the secret key k , initial value IV , sequence⁴ $\mathbf{v}_t = \{\psi_d^t(r_0)\}_{t=l_0+1}^{l_0+q}$ and its sorted sequence $\mathbf{x}_i = \{\psi_d^i(r_0)\}_{i=1}^q$, and returns a key permutation π_k , an initial state vector $\mathbf{s}_{-\iota}$ and an initial memory vector $\tilde{\mathbf{c}}_{-\iota}$. To generate the chaotic permutation π_k , we follow [2] and define,

$$\pi_k(i) = \begin{cases} j & \psi_d(x_i) = x_j \\ j' & (v_{q-1} = x_i \text{ and } v_0 = x_{j'}). \end{cases} \quad (2.1.3)$$

Unfortunately, this pseudorandom permutation is not by itself sufficient for image encryption mainly because of correlations existing in an image, most of which concentrated in high frequency components (e.g. see Figure 1).

2.2 The kernel computation phase

This phase contains state update algorithm and key stream generator as shown in Equations (2.2.1), (2.2.2) and (2.2.3). In the next paragraph, synchronous and self-synchronous modes

⁴Here we may assume that the vector \mathbf{v}_t does not have repeated entries by chaotic properties of the Renyi map. Clearly since the probability of having a bad vector with two identical entries is negligible one may eventually find a good vector by repeating the process.

are explained. As one may note, both modes have similar chaotic maps, internal states and permutation-substitution components, but the self-synchronous mode has an internal memory and a memory update function in order to synchronize both transmitter and receiver simultaneously. The memory update function μ is defined as follows,

$$\mu(\tilde{\mathbf{c}}_t, \mathbf{c}_{t-1}) \stackrel{\text{def}}{=} \mathbf{M}\tilde{\mathbf{c}}_t + \mathbf{c}_{t-1}$$

in which

$$\tilde{\mathbf{c}}_t = [\tilde{c}_t^{(\ell)}, \tilde{c}_t^{(\ell-1)}, \dots, \tilde{c}_t^{(2)}, \tilde{c}_t^{(1)}]^T, \quad \tilde{c}_t^{(i)} = c_{t-\ell+i-1},$$

$$\mathbf{M} \stackrel{\text{def}}{=} \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 1 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \end{bmatrix}.$$

The state update algorithm gets a current state \mathbf{s}_t and returns the next state \mathbf{s}_{t+1} . Clearly, as in (2.2.1), the state update algorithm consists of a linear part and a chaotic permutation. The linear part not only connects a relation between the internal state and previous cipher symbols, but also increases shuffling property which results in an almost perfect uniformly distributed output. The keystream generator algorithm gets the current state \mathbf{s}_t and returns the keystream \mathbf{z}_t .

Let $\mathbf{A}, \mathbf{B}, \mathbf{D}, \mathbf{E}, \mathbf{F}, \mathbf{W} \in \mathcal{M}$ while \mathbf{F} is invertible. The kernel of PLCIE in the synchronous mode, Kernel_s , is defined as follows.

$$\text{Kernel}_s : \begin{cases} \text{initial} & : \pi_k, \mathbf{A}, \mathbf{B}, \mathbf{D}, \\ \mathbf{s}_{t+1} & = \mathbf{D}\mathbf{s}_t + \mathbf{A}_{\wp_k}(\mathbf{s}_t) \\ \mathbf{z}_t & = \mathbf{B}_{\wp_k}(\mathbf{s}_t) \end{cases} \quad (2.2.1)$$

To control the error diffusion rate, correct state recovery and maintain stability, one may improve the above kernel to work in a self-synchronized mode as follows,

$$E\text{Kernel}_{ss} : \begin{cases} \text{initial} & : \pi_k, \mathbf{A}, \mathbf{B}, \mathbf{D}, \mathbf{E}, \mathbf{W} \\ \mathbf{s}_{t+1} & = \mathbf{W}\tilde{\mathbf{c}}_t + \mathbf{D}\mathbf{s}_t + \mathbf{A}_{\wp_k}(\mathbf{s}_t) + \mathbf{E}_{\wp_k}(p_t) \\ \tilde{\mathbf{c}}_{t+1} & = \mathbf{M}\tilde{\mathbf{c}}_t + \mathbf{c}_{t-1} \\ \mathbf{z}_t & = \mathbf{W}\tilde{\mathbf{c}}_t + \mathbf{B}_{\wp_k}(\mathbf{s}_t), \end{cases} \quad (2.2.2)$$

Note that one get the synchronous mode when $\mathbf{W} = \mathbf{E} = \mathbf{0}$. It is proved in [12] (see Theorem A below) that if

- 1- $\mathbf{A} = \mathbf{E}\mathbf{F}^{-1}\mathbf{B}$,
- 2- there exists $n_0 \in \mathbb{N}$ such that $\mathbf{D}^{n_0} = \mathbf{0}$

then $E\text{Kernel}_{ss}$ has an (unknown input) observer defined as

$$D\text{Kernel}_{ss} : \begin{cases} \text{initial} & : \pi_k, \langle \mathbf{c}_t \rangle, \mathbf{A}, \mathbf{B}, \mathbf{D}, \mathbf{E}, \mathbf{F}, \mathbf{W} \\ \hat{\mathbf{s}}_{t+1} & = \mathbf{W}\tilde{\mathbf{c}}_t + \mathbf{D}\hat{\mathbf{s}}_t + \mathbf{A}_{\wp_k}(\hat{\mathbf{s}}_t) + \mathbf{E}\mathbf{F}^{-1}(\mathbf{c}_t - \hat{\mathbf{z}}_t) \\ \tilde{\mathbf{c}}_{t+1} & = \mathbf{M}\tilde{\mathbf{c}}_t + \mathbf{c}_{t-1} \\ \hat{\mathbf{z}}_t & = \mathbf{W}\tilde{\mathbf{c}}_t + \mathbf{B}_{\wp_k}(\hat{\mathbf{s}}_t). \end{cases} \quad (2.2.3)$$

2.3 The encryption/decryption phase

Based on the kernel equations we have the following procedures for encryption and decryption in general,

$$\text{Enc} : \begin{cases} \text{input} : \langle p_t \rangle \\ \text{EKernel} \\ \text{output} : \mathbf{c}_t = \mathbf{z}_t + \mathbf{F}_{\wp_k}(\mathbf{p}_t). \end{cases} \quad (2.3.1)$$

$$\text{Dec} : \begin{cases} \text{input} : \langle c_t \rangle \\ \text{DKernel} \\ \text{output} : \hat{\mathbf{p}}_t = \wp_k^{-1}(\mathbf{F}^{-1}(\mathbf{c}_t - \hat{\mathbf{z}}_t)). \end{cases} \quad (2.3.2)$$

Note that in this setting the matrix \mathbf{E} is secret and is included in the key. The vector IV is chosen at random and is sent along with the ciphertext to make sure that a trivial CPA attack is not applicable. Moreover, if one is not working in a lightweight setting then one may also encode the matrix \mathbf{B} in the key and make it secret to enhance security conditions of the scheme.

Based on the following theorem [12], by making \mathbf{A} secret as a function of the key and choosing \mathbf{D} properly, one may prove that a receiver as an UIO exists. We recall the result along with a sketch of proof for the scheme as follows.

Theorem A. *In PLCIE , if*

- a) $\mathbf{A} = \mathbf{E}\mathbf{F}^{-1}\mathbf{B}$,
- b) *The matrix \mathbf{D} is nilpotent i.e. there exists an integer $n_0 \in \mathbb{N}$ such that $\mathbf{D}^{n_0} = \mathbf{0}$,*

then

- I. $\forall t, t \in \{1, 2, \dots, n_0\}$, \mathbf{s}_{t+1} *depends on the $\ell + t$ previous cipher symbols.*
- II. $\forall t, t \in \{n_0 + 1, n_0 + 2, \dots\}$, \mathbf{s}_{t+1} *depends on the $\ell + n_0$ previous cipher symbols.*
- III. *After n_0 time step, an unknown input observer can detect correct plain symbols.*

Sketch of proof.

For (I), at first, by induction on $t \geq 1$ we prove an equivalent explicit form of the internal state \mathbf{s}_{t+1} as follow,

$$\mathbf{s}_{t+1} = \sum_{j=1}^t \mathbf{D}^{j-1} [(\mathbf{I} - \mathbf{E}\mathbf{F}^{-1})\mathbf{W}\tilde{\mathbf{c}}_{t-j+1} + \mathbf{E}\mathbf{F}^{-1}\mathbf{c}_{t-j+1}] + \mathbf{D}^t \mathbf{s}_1. \quad (2.3.3)$$

Now by definition 2.2 of $\tilde{\mathbf{c}}_t$ we have,

$$\tilde{\mathbf{c}}_t = [\tilde{c}_t^{(\ell)}, \tilde{c}_t^{(\ell-1)}, \dots, \tilde{c}_t^{(2)}, \tilde{c}_t^{(1)}]^T \stackrel{\text{def}}{=} [c_{t-1}^{(1)}, c_{t-2}^{(1)}, \dots, c_{t-\ell}^{(1)}]^T$$

and consequently, for all $1 \leq j \leq t$, we can write $\tilde{\mathbf{c}}_{t-j+1}$ as follow,

$$\tilde{\mathbf{c}}_{t-j+1} = [c_{t-j}^{(1)}, c_{t-j-1}^{(1)}, \dots, c_{t-j+1-\ell}^{(1)}]^T,$$

proving part (I).

For (II), suppose that there exist $n_0 \in \mathbb{N}$ such that $\mathbf{D}^{n_0} = \mathbf{0}$. Then expand \mathbf{s}_{t+1} for $t = n_0 + 1$ as,

$$\begin{aligned}
\mathbf{s}_{t+1} = & \mathbf{D}^0[(\mathbf{I} - \mathbf{EF}^{-1})\mathbf{W}\tilde{\mathbf{c}}_{n_0+1} + \mathbf{EF}^{-1}\mathbf{c}_{n_0+1}] + \\
& \mathbf{D}^1[(\mathbf{I} - \mathbf{EF}^{-1})\mathbf{W}\tilde{\mathbf{c}}_{n_0} + \mathbf{EF}^{-1}\mathbf{c}_{n_0}] + \\
& \vdots \\
& \mathbf{D}^{n_0-1}[(\mathbf{I} - \mathbf{EF}^{-1})\mathbf{W}\tilde{\mathbf{c}}_2 + \mathbf{EF}^{-1}\mathbf{c}_2] + \\
& \mathbf{D}^{n_0}[(\mathbf{I} - \mathbf{EF}^{-1})\mathbf{W}\tilde{\mathbf{c}}_1 + \mathbf{EF}^{-1}\mathbf{c}_1] + \\
& \mathbf{D}^{n_0+1}\mathbf{s}_1,
\end{aligned}$$

and since $\mathbf{D}^{n_0} = \mathbf{0}$, for any $\nu \geq 1$ and an arbitrary state $\mathbf{s}_{n_0+\nu}$ we have,

$$\begin{aligned}
\mathbf{s}_{n_0+\nu} = & \mathbf{D}^0[(\mathbf{I} - \mathbf{EF}^{-1})\mathbf{W}\tilde{\mathbf{c}}_{n_0+\nu-1} + \mathbf{EF}^{-1}\mathbf{c}_{n_0+\nu-1}] + \\
& \mathbf{D}^1[(\mathbf{I} - \mathbf{EF}^{-1})\mathbf{W}\tilde{\mathbf{c}}_{n_0+\nu-2} + \mathbf{EF}^{-1}\mathbf{c}_{n_0+\nu-2}] + \\
& \vdots \\
& \mathbf{D}^{n_0-1}[(\mathbf{I} - \mathbf{EF}^{-1})\mathbf{W}\tilde{\mathbf{c}}_\nu + \mathbf{EF}^{-1}\mathbf{c}_\nu] + \\
& \stackrel{\text{def}}{=} \phi_2(\mathbf{c}_{\nu-\ell}^{(1)}, \dots, \mathbf{c}_{n_0+\nu-1}^{(1)}),
\end{aligned} \tag{2.3.4}$$

proving (II).

For (III), define $\mathbf{e}_{t+1} \stackrel{\text{def}}{=} \mathbf{s}_{t+1} - \hat{\mathbf{s}}_{t+1}$ and note that by 2.2.2 and 2.2.3 we have,

$$\begin{aligned}
\mathbf{e}_{t+1} &= \mathbf{s}_{t+1} - \hat{\mathbf{s}}_{t+1} \\
&= \mathbf{W}\tilde{\mathbf{c}}_t + \mathbf{D}\mathbf{s}_t + \mathbf{A}_{\wp_k}(\mathbf{s}_t) + \mathbf{E}_{\wp_k}(\mathbf{p}_t) - \\
&\quad (\mathbf{W}\tilde{\mathbf{c}}_t + \mathbf{D}\hat{\mathbf{s}}_t + \mathbf{A}_{\wp_k}(\hat{\mathbf{s}}_t) + \mathbf{EF}^{-1}(\mathbf{c}_t - \hat{\mathbf{z}}_t)) \\
&= \mathbf{D}\mathbf{e}_t + \mathbf{A}_{\wp_k}(\mathbf{s}_t) + \mathbf{E}_{\wp_k}(\mathbf{p}_t) - \\
&\quad \mathbf{A}_{\wp_k}(\hat{\mathbf{s}}_t) - \mathbf{EF}^{-1}(\mathbf{c}_t - \hat{\mathbf{z}}_t).
\end{aligned} \tag{2.3.5}$$

Using 2.2.2 and 2.2.3 one may conclude that,

$$\begin{aligned}
\mathbf{e}_{t+1} &= \mathbf{D}\mathbf{e}_t + \mathbf{A}_{\wp_k}(\mathbf{s}_t) - \mathbf{EF}^{-1}(\mathbf{W}\tilde{\mathbf{c}}_t + \mathbf{B}_{\wp_k}(\mathbf{s}_t) - \mathbf{W}\tilde{\mathbf{c}}_t) \\
&= \mathbf{D}\mathbf{e}_t + \mathbf{A}_{\wp_k}(\mathbf{s}_t) - \mathbf{EF}^{-1}(\mathbf{B}_{\wp_k}(\mathbf{s}_t)),
\end{aligned} \tag{2.3.6}$$

and consequently,

$$\mathbf{e}_{t+1} = \mathbf{D}\mathbf{e}_t, \tag{2.3.7}$$

that proves (III). ■

3 Performance analysis

In this section, we concentrate on the performance and statistical evaluation of our proposed scheme PLCIE .

First, let us consider the performance of the scheme in general and most importantly in lightweight setups. Since, to the best of our knowledge, there is no self-synchronous image encryption scheme similar to our proposed scheme, we have decided to compare our scheme with Moustique [4] which is one of the fastest proposed self-synchronous stream cipher existing so far⁵ [10].

⁵Although there exists severe attacks to Moustique (e.g. see [11]), we have chosen this scheme since we are not aware of any better self-synchronized stream cipher similar to what we have proposed.

In this regard, consider a $4N$ -bit input plaintext given to both systems. To generate the ciphertext, the number of field operations for Moustique is $6000N$. On the other hand, for PLCIE we may consider two sets of parameters which are comparable with Moustique, namely ($prec = 16$, $\ell = 8$, $n = 6$ and $\mathbb{F}_q = GF(16)$) with key length 97 and ($prec = 32$, $\ell = 8$, $n = 5$ and $\mathbb{F}_q = GF(16)$) with key length 119. In both of these setups the number of field operations to produce the ciphertext given a $4N$ -bit plaintext is $528N$ which shows that PLCIE is about 11 times faster than Moustique in bit production. Of course one should also note that in our setup we use a bandwidth 8 times more than Moustique, which give rise to an over-all speed factor of 1.5 in favor of PLCIE .

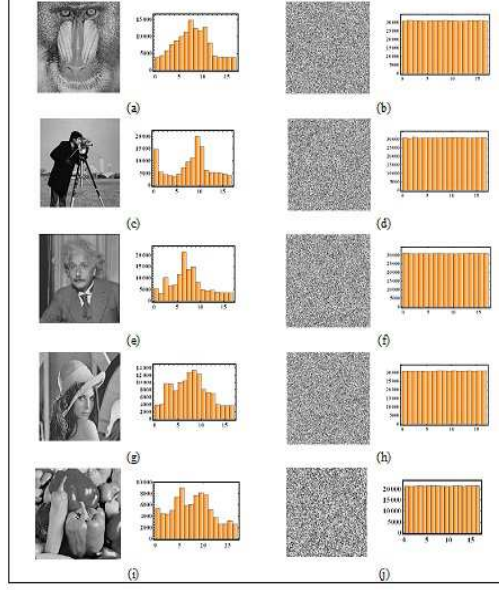


Figure 2: Images (a),(c),(e),(g),(i) depict histograms of original images and images (b),(d),(f),(h),(j) depict histograms of encrypted images.

To make sure about the uniformity of the output distribution first refer to Figure 2 that shows the histograms of some original standard gray images and their corresponding encrypted images, showing an almost uniformly distributed outputs. In order to be more precise, we have used NIST Sp-800 Suite [18] tests for 40 binary sequences of cipher images with 1000,000 bit length, generated for different secret keys. As it is reported in Figure 3 for the Peper image below, PLCIE passes all these tests with an acceptable confidence interval.

On the other hand, in order to test the influence of changing a single symbol in the original image on the encrypted image, the number of symbols' change rate is measured by calculating NPCR (number of symbol change rate) and UACI (unified average changing intensity) as follows (e.g. see [20] for more on these standard parameters),

$$NPCR = \frac{100}{W \times H} \sum_{i,j} D(i,j) \quad , \quad UACI = \frac{100}{W \times H} \sum_{i,j} \frac{|C(i,j) - C'(i,j)|}{q-1}$$

in which W and H are the width and the height of encrypted images. Note that NPCR measures the percentage of different symbols between the two cipher images and UACI measures the average intensity of differences between the two cipher images. Two encrypted images C and C' , whose corresponding original images P and P' have only one-symbol difference, are considered. A two-dimensional array D with the same size of C and C' is defined for

Test Name	Average	χ^2_{prop}	Result
Frequency	%100.00	0.3939	Pass
Frequency within a Block	%100.00	0.3939	Pass
Runs	%100.00	0.3939	Pass
Longest Run of Ones in a Block	%97.44	0.9637	Pass
Binary Matrix Rank	%97.44	0.9637	Pass
Discrete Fourier Transform	%100.00	0.3939	Pass
Non Overlapping Template Matching	%97.44	0.9637	Pass
Overlapping Template Matching	%100.00	0.3939	Pass
Maurer	%94.87	6.7135	Pass
Linear Complexity	%100.00	0.3939	Pass
Serial	%100.00	0.3939	Pass
Approximate Entropy	%100.00	0.3939	Pass
Cumulative Sums Forward	%100.00	0.3939	Pass
Cumulative Sums Backward	%100.00	0.3939	Pass
Random Excursions	%100.00	0.3939	Pass
Random Excursions Variant	%100.00	0.3939	Pass

Figure 3: Randomness of binary sequences for Peper cipher image

which $D(i, j) = 1$ if $C(i, j) \neq C'(i, j)$, and $D(i, j) = 0$ otherwise. As Table 2 reflects our experimental results, PLCIE has good cipher image sensibility to little perturbation in plain images.

Also, the cipher image dependency on a small perturbation in secret key bits is analyzed. Table 3 shows the detailed results for the encryption of the Peper image with two secret keys, which have just a one bit difference.

3.1 Cipher image entropy and correlation analysis

Information entropy is one of the most significant features of randomness. Information entropy $h(m)$ of a message m can be measured by the following formula,

$$h(m) = - \sum_{i=0}^{n-1} p(m_i) \log_2(p(m_i)), \quad (3.1.1)$$

where n is the total number of symbols in the message m and $p(m_i)$ represents the probability of the occurrence of symbol m_i . Theoretically, for a random code source with an alphabet of size n , the ideal information entropy must be $h(m) = \log_2(n)$. Table 4 shows that in PLCIE the entropy of encrypted images for four standard images are close to ideal values, which shows robustness against entropy attacks.

Also, there is usually strong correlations between adjacent symbols in the input image. A secure image encryption scheme should remove this correlation to make statistical attacks infeasible. In order to test the correlation between adjacent symbols, 2500 random pairs of adjacent symbols (in horizontal, vertical, and diagonal directions) are selected and the correlation coefficient of each pair is computed before and after encryption using the following equations,

Table 2: Plain image sensitivity analysis

	<i>NPCR</i>	<i>UACI</i>
Baboon	99.552	33.171
Camera Man	99.572	33.239
Einstein	99.543	33.229
Lena	99.540	33.239
Peper	99.597	33.250

Table 3: Key sensitivity analysis

	<i>NPCR</i>	<i>UACI</i>
Peper	99.549	33.219

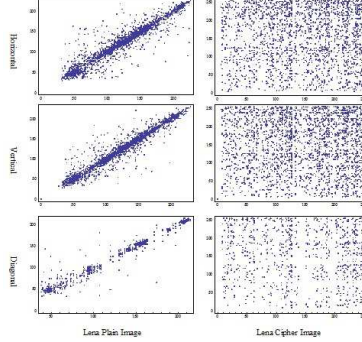


Figure 4: Correlation plot of adjacent symbols in the Lena plain and cipher images

$$Cor = \frac{\sum_{i=1}^N (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\left(\sum_{i=1}^N (x_i - \bar{x})^2\right) \left(\sum_{i=1}^N (y_i - \bar{y})^2\right)}} \quad (3.1.2)$$

where \bar{x} and \bar{y} are average values. The correlation plot of the plain image and the cipher image of Lena is illustrated in Figure 4. Also, Table 5 summarizes the results corresponding to 4 other images.

3.2 Self-synchronizing property analysis

Another feature of PLCIE is the fact that the scheme can be used in a self-synchronous mode maintaining error correction using an unknown input observer as a receiver (e.g. see [15] for more on this method). To show this property, Figures 5 and 6 demonstrate the result of and error recovery in the Baboon image where the experiment are depicted numerically and graphically in these figures.

Table 4: Byte, symbol and bit entropy analysis of plain and cipher images

	<i>Bit entropy</i>		<i>Symbol entropy</i>		<i>Byte entropy</i>	
	<i>Plain</i>	<i>Cipher</i>	<i>Plain</i>	<i>Cipher</i>	<i>Plain</i>	<i>Cipher</i>
Baboon	0.84	0.97	3.92	4.0	7.33	7.69
Camera Man	0.77	0.97	3.85	4.0	7.01	7.69
Einstein	0.78	0.97	3.83	4.0	6.88	7.68
Lena	0.79	0.97	3.94	4.0	7.44	7.68
Peper	0.77	0.97	3.98	4.0	7.59	7.69

Table 5: Correlation analysis of plain and cipher images

	<i>Plain</i>			<i>Cipher</i>		
	<i>Hori.</i>	<i>Vert.</i>	<i>Diag.</i>	<i>Hori.</i>	<i>Vert.</i>	<i>Diag.</i>
Baboon	0.669	0.723	0.643	0.019	0.034	0.005
Camera Man	0.935	0.976	0.913	0.006	0.011	0.019
Einstein	0.892	0.723	0.912	0.027	0.034	0.006
Lena	0.910	0.961	0.913	0.035	0.019	0.001

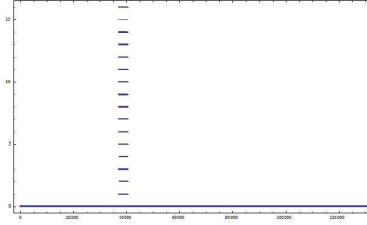


Figure 5: Numerical difference of sent cipher image data and received the one.

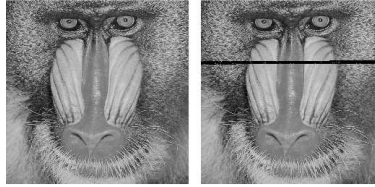


Figure 6: Error recovery in the self-synchronous mode for Baboon.

References

- [1] ADDABBO, T., A. FORT, S. ROCCHI AND V. VIGNOLI, *Digitized chaos for pseudo-random number generation in cryptography*, In Chaos-Based Cryptography, Eds. L. Kocarev and S. Lian, Springer-Verlag Berlin, (2011) 67-97.
- [2] AMIGÓ, J. M., J. SZCZEPANSKI AND L. KOCAREV, *Discrete chaos and cryptography*, in Proceedings of International Symposium on Nonlinear Theory and its Applications (NOLTA2005), Bruges, Belgium, October 18-21, (2005) 461-464.
- [3] AMIGÓ, J. M., L. KOCAREV AND J. SZCZEPANSKI, *Theory and practice of chaotic cryptography*, Physics Letters A, **366** (2007) 211-216.
- [4] DAEMEN, J. AND P. KITOS, *The Self-synchronizing stream cipher Moustique*, in New Stream Cipher Design, Eds. M. Robshaw and O. Billot, LNCS 4986, Springer-Verlag Berlin, (2008) 210 -223.
- [5] DANESHGAR, A. AND F. MOHEBBIPOOR, *A switching chaotic stream cipher*, (2014) (manuscript).
- [6] DEVANEY, R. L., *An Introduction to chaotic dynamical systems*, 2nd Ed., Westview Press, (2003) 340 pp.
- [7] FARAGALLAH, O. S., *Efficient confusion/diffusion chaotic image cryptosystem using enhanced standard map*, Signal, Image and Video Processing, Oct. (2014) online, DOI: 10.1007/s11760-014-0683-y.
- [8] FRIDRICH, J., *Image encryption based on chaotic maps*, IEEE International Conference on Computational Cybernetics and Simulation, **Vol. 2** (1997).
- [9] GALATOLO, S., H. MATHIEU AND R. CRISTÓBAL, *Statistical properties of dynamical systems-simulation and abstract computation*, Chaos Solitons and Fractals, **45**, (2012) 1-14.

- [10] GOOD, T. AND M. BENAÏSSA, 2006, Hardware performance of eSTREAM phase-III stream cipher candidates, SASC 2006.
- [11] KASPER, E., V. RIJMEN, T. BJØRSTAD, C. RECHBERGER, M. ROBSHAW AND G. SEKAR, *Correlated keystreams in Moustique*, in Progress in Cryptology - AFRICACRYPT 2008, Ed. S. Vaudenay, LNCS 5023, Springer-Verlag Berlin, (2008) 246-257.
- [12] KHADEM, B., A. DANESHGAR AND F. MOHEBPOUR, *A stream cipher based on chaotic permutations*, Kharazmi University Journal of Sciences, (to appear) (in Persian).
- [13] KWOK, H. S. AND K. S. T. WALLACE, *A fast image encryption system based on chaotic maps with finite precision representation*, Chaos Solitons and Fractals, **32** (2007) 1518-1529.
- [14] LI, C., S. LI, G. CHEN AND W. A. HALANG, *Cryptanalysis of an image encryption scheme based on a compound chaotic sequence*, Image and Vision Computing, **Vol. 27** (2009) 1035-1039.
- [15] MILLÉRIEUX, G., J. M. AMIGÓ AND J. DAAFOUZ, *A connection between chaotic and conventional cryptography*, IEEE Transactions on Circuits and Systems I, **55** (2008) 1695-1703.
- [16] MINTU, P. AND A. DAS, *Survey of image encryption using chaotic cryptography schemes*, IJCA Special Issue on Computational Science-New Dimensions and Perspectives, NCCSE (2011) 1-4.
- [17] MISRA, A., A. GUPTA AND D. RAI, *Analysing the parameters of chaos based image encryption schemes*, World Applied Programming, **1** (2011) 294-299.
- [18] RUKHIN, A. AND COAUTHORS, *A statistical test suite for random and pseudorandom number generators for cryptographic applications*, Booz Allen Hamilton Inc., Mclean, VA, (2001).
- [19] SHARMA, M. AND M. K. KOWAR, *Image encryption techniques using chaotic schemes: a review*, International Journal of Engineering Science and Technology, **Vol. 2**, (2010).
- [20] SU, Z., G. ZHANG AND J. JIANG, *Multimedia security: a survey of chaos-based encryption technology*, in Multimedia: A Multidisciplinary Approach to Complex Issues, Ed. I. Karydis, InTech, (2012) 99-124.
- [21] TONG, X. AND M. CUI, *Image encryption scheme based on 3D baker with dynamical compound chaotic sequence cipher generator*, Image and Vision Computing, **Vol. 26** (2008) 843-850.